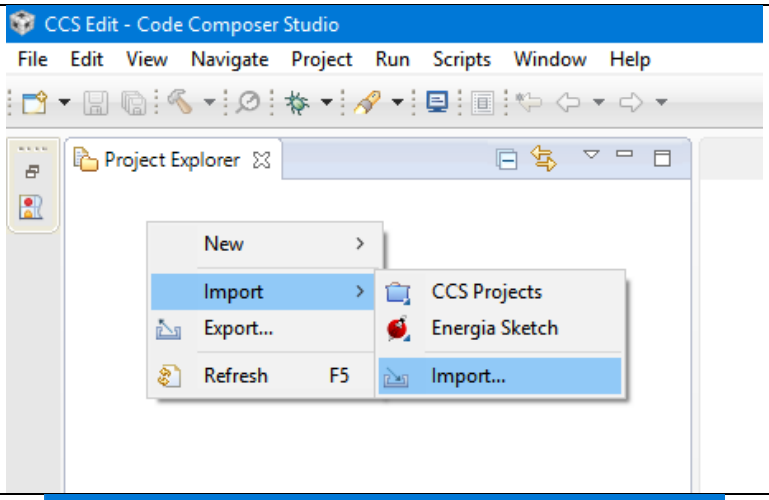
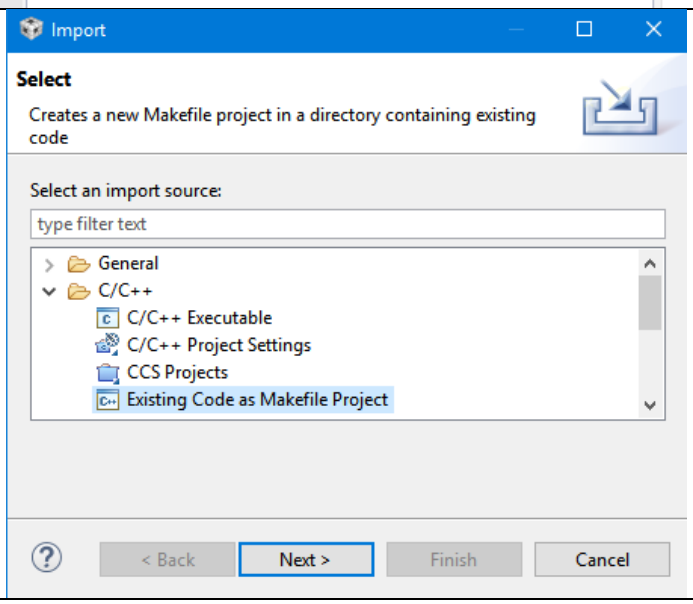
How to Flash & Debug an Ada application

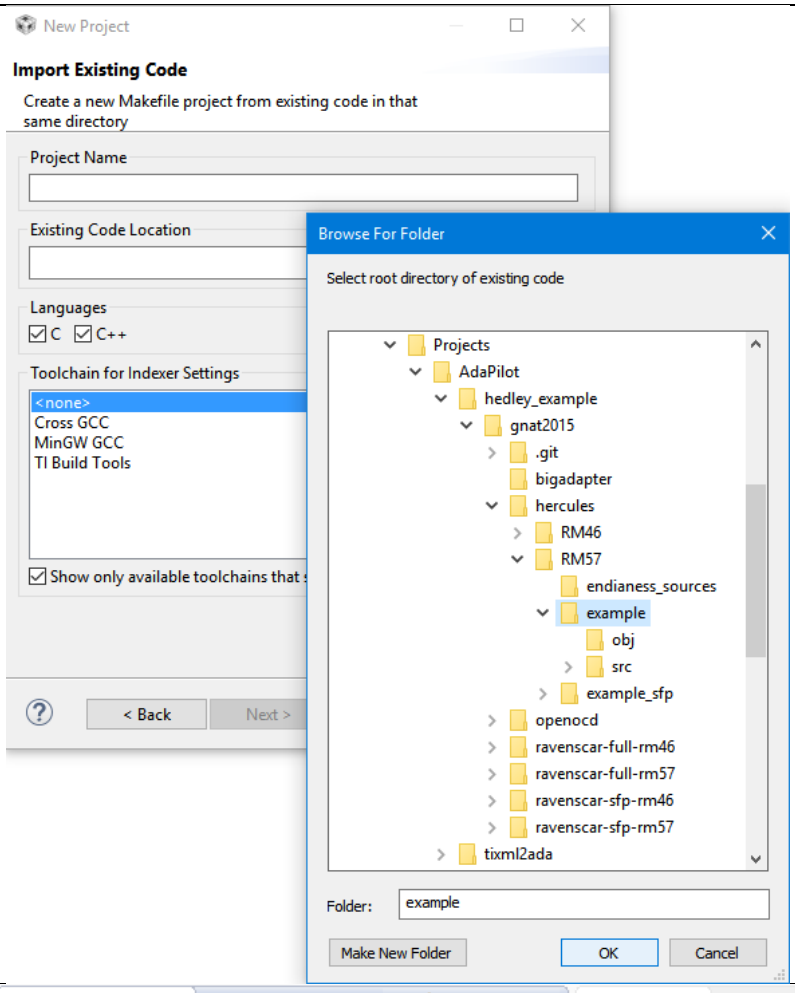
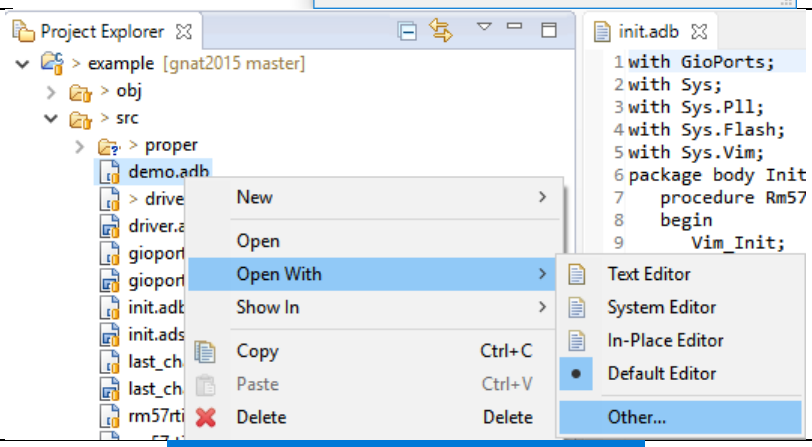
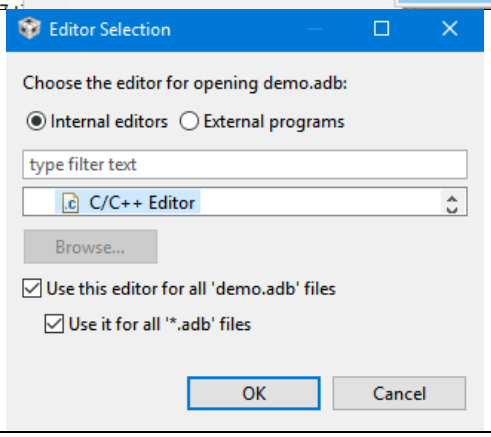
on the TI TMS570LC43xx Launchpad

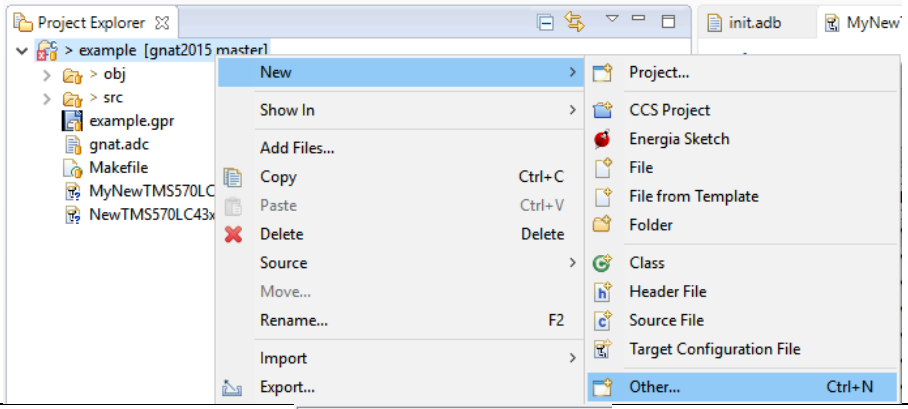
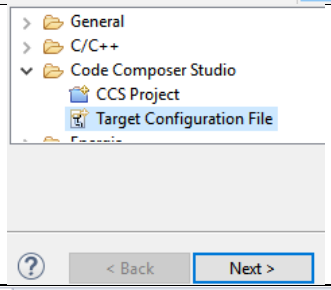
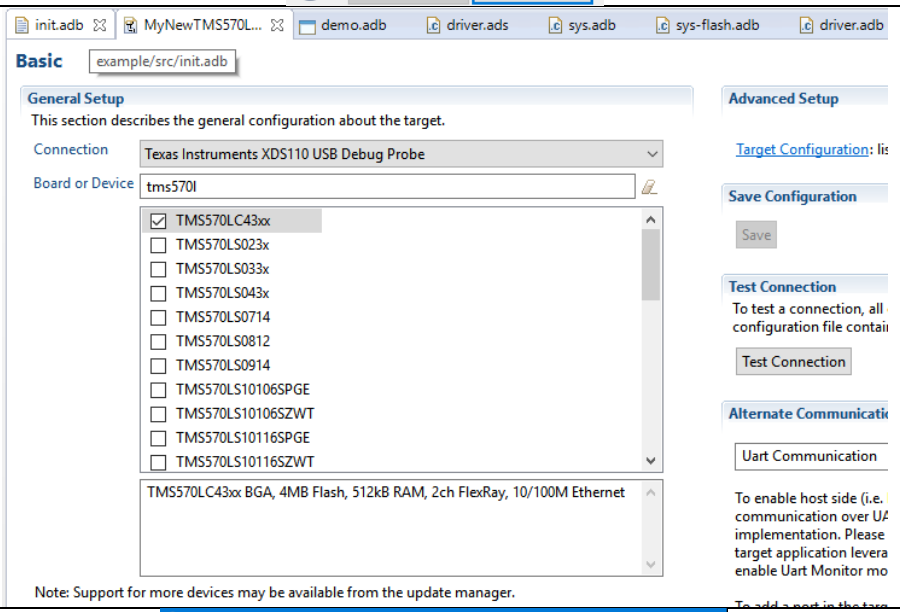
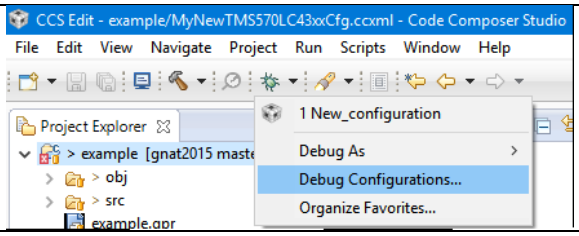
using Code Composer and the XDSv110 probe

Project AdaPilot
Release date 02 Nov 2016
Version 0.1

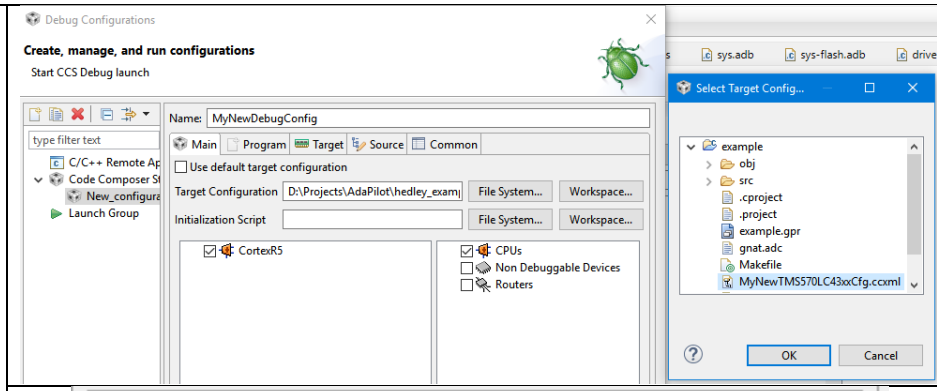
This document describes how to get started on debugging an application compiled with the Ada GNAT compiler on the TI TMS570LC43x Launchpad. This method doesn't require a separate debugger – it works with the on-board XDSv110 probe and TI's Code Composer Studio IDE, both free for non-commercial use.
A simple HelloWorld program is available here: **https://github.com/PastravMD/gnat2015**

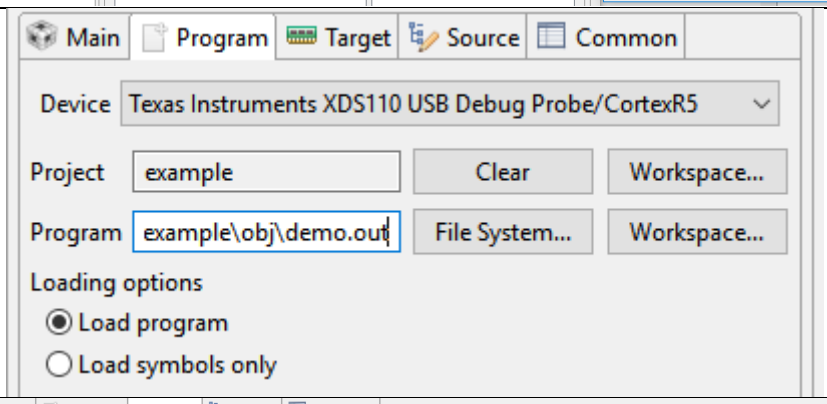
1	**Download free copy of Code Composer Studio**	http://www.ti.com/tool/ccstudio
2	**Install CCS with Hercule (Cortex R) support**	
3	**Import the Ada Project folder in Eclipse:** a. right-click in Project Explorer and select Import b. Import "Existing Code as Makefile Project" c. Select your Project Folder and click ok	 

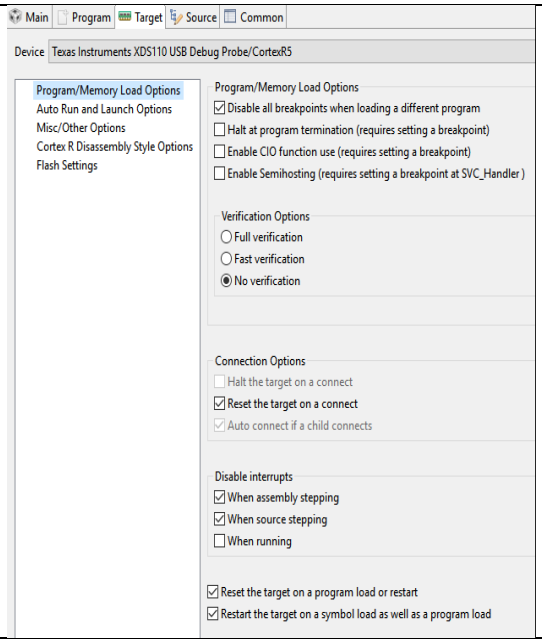
		
4	**Set CCS to open the .adb and .ads source files with the standard text editor:** a. Right-click a .adb file from the project and select Open With > Other… b. Select “C/C++ Editor” c. Check both “Use this editor for all .adb files” checkboxes. d. Click OK	
		

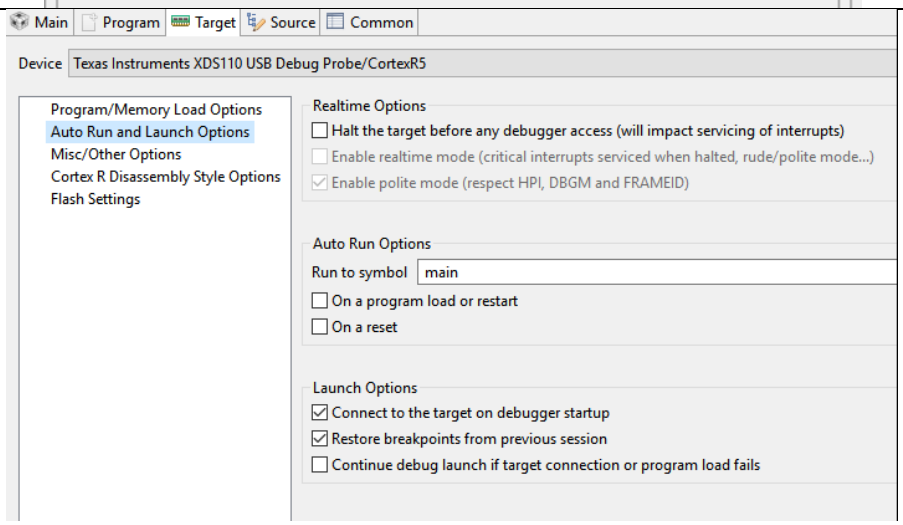
5	**Create a CCS Target Configuration File:** a. right-click on the project folder and select New > Other … b. From the Code Composer Studio category, select Target Configuration File c. Give it a fitting name	 
6	**Set the Target Configuration File:** a. Open the new .ccxml file in Code Composer b. Select the XDSv110 USB Debug Probe c. Select the TMS570LC43xx Device d. Save the configuration e. Press the Test button f. In the test report pop-up window, you should be able to find 3 successful reports: *This utility has successfully reset the controller.* … *The JTAG IR Integrity scan-test has succeeded.* … *The JTAG DR Integrity scan-test has succeeded.*	
7	**Create CCS Project Debug Configuration:** a. Click the Debug Menu Button > Debug Configurations … b. Create a new CCS Debug Configuration	

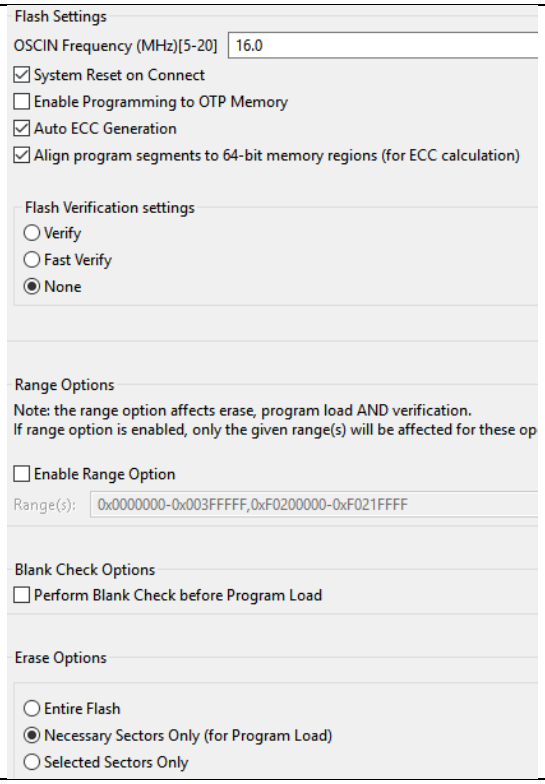
c. Under the ‘**Main**’ tab, in the Target Configuration field, select the CCS Target Cfg file you created previously.
d. Under the ‘Program’ tab select your project from workspace and indicate which file is your binary ELF file that’s going to be flashed and debugged.
e. In the ‘Target’ tab set you debugging preferences. A set of working values can be seen in the screenshots.
f. In Target > Flash Settings > Erase Options, selecting “Necessary Sectors Only” will make flashing much faster.
g. Source & Common tabs are ok as default
h. Click Apply & Debug

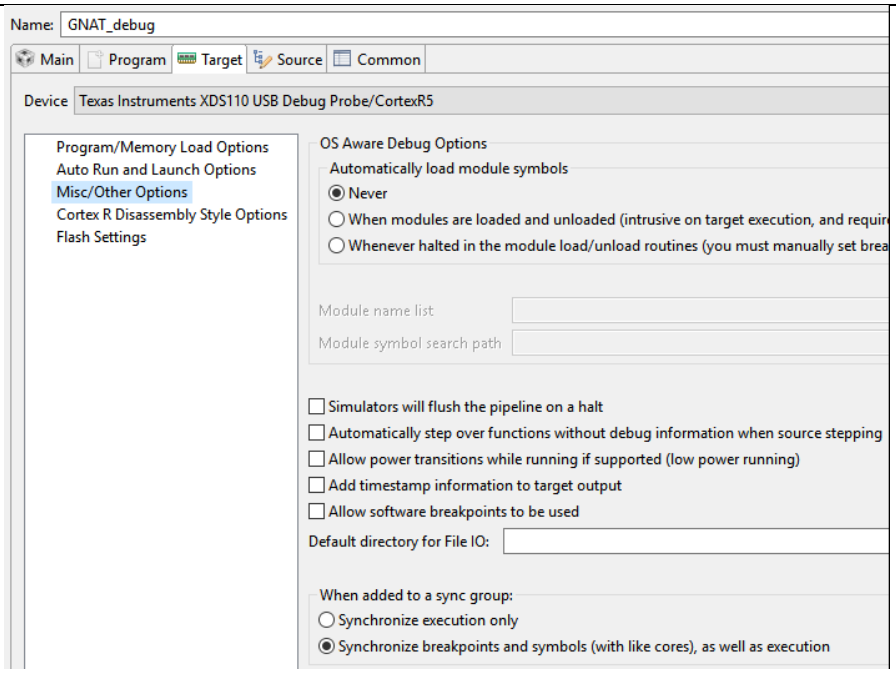






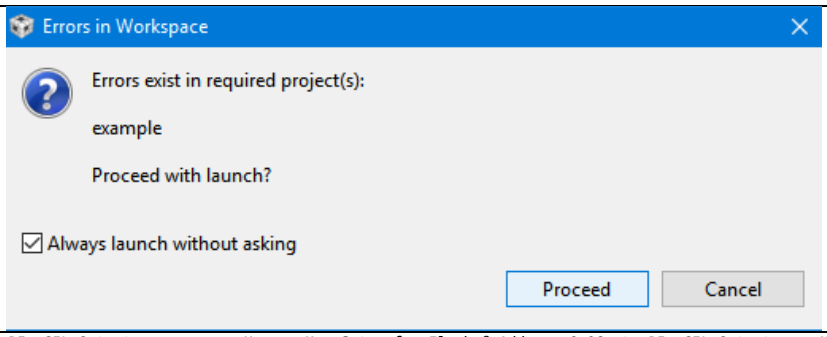






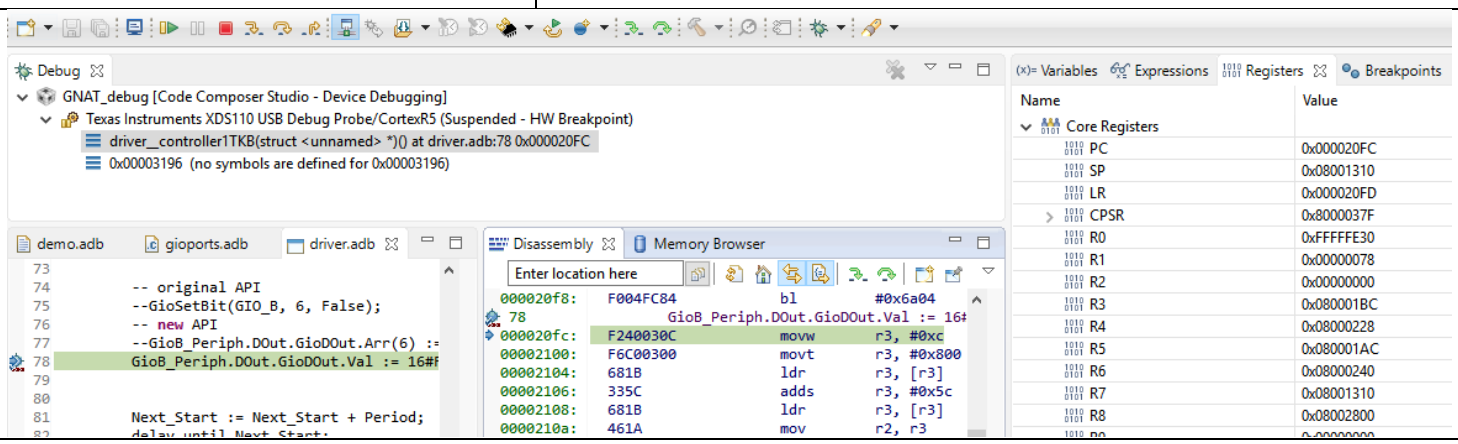
8 Debug the embedded application:

a. When launching the newly created debug configuration, ignore the reported project error.
b. The ELF file will be flashed automatically and you should see a similar report in the Console window.
c. MCU should now be reset and halted immediately after.
d. All standard Eclipse debugging practices should apply from here on.



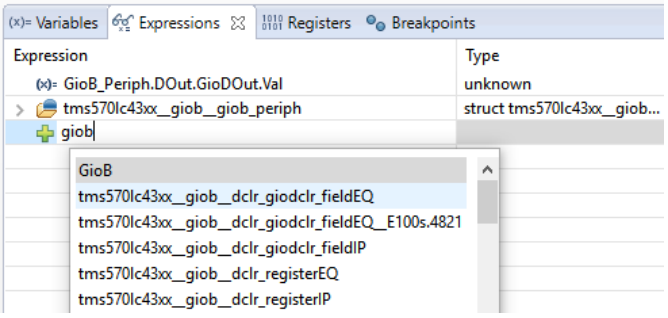

```
CortexR5: GEL Output:            Memory Map Setup for Flash @ Address 0x0CortexR5: GEL Output:     Memory
Map Setup for Flash @ Address 0x0 due to System Reset
CortexR5: GEL Output:            Memory Map Setup for Flash @ Address 0x0 due to System Reset
CortexR5: Writing Flash @ Address 0x00000000 of Length 0x00007ff0
CortexR5: Erasing Flash Bank 0, Sector 0
CortexR5: Erasing Flash Bank 0, Sector 1
CortexR5: Writing Flash @ Address 0x00007ff0 of Length 0x00007268
CortexR5: Erasing Flash Bank 0, Sector 2
CortexR5: Erasing Flash Bank 0, Sector 3
CortexR5: GEL Output:            Memory Map Setup for Flash @ Address 0x0 due to System Reset
```

9



10 Shortcomings:

a. CCS doesn't support the Ada language by default so there's no syntax highlighting in the editor
b. CCS can't properly identify the variable symbols generated by the GNAT compiler
   ➔ To debug variables & data, search for the symbol name in the "**Expressions**" View



11 **Improved register descriptions for Code Composer:**

Since the CCS register descriptions for the TMS570 device are also used to generate the Ada spec files, various improvements and corrections were applied to the original files.
These improvements can also help make debugging in CCS easier if the target description XML files are imported back into CCS (see comparison fig below).

To do this:
a. clone the code generator repository, where the updated xml files are kept:
   https://github.com/PastravMD/tixml2ada.git
b. Copy all contents of folder **tixml2ada\input\Modules\hercules**
   and paste & overwrite **CodeComposer_path\ccsv6\ccs_base\common\targetdb\Modules\hercules**
   (a backup of the originals is a good idea)

Original reg description:

GioB		
Dir	0x000000C0	Data Direction Gio B [Memory Mapped]
DIn	0x00000030	Data Input Gio B [Memory Mapped]
DOut	0x00000000	Data Output Gio B [Memory Mapped]
DSet	0x00000000	Data Set Gio B [Memory Mapped]
DClr	0x00000000	Data Clear Gio B [Memory Mapped]
PDr	0x00000000	Open Drain Gio B [Memory Mapped]
PDis	0x00000000	Pull Disable Gio B [Memory Mapped]
PSel	0x00000000	Pull Select Gio B [Memory Mapped]

Enriched Description:

GioB		
Dir	0x00000000	Data Direction Gio B [Memory Mapped]
Reserved	000000000000000000000000	Read returns 0. Writes have no effect.
GioDir7	0	GIO data direction of port n
GioDir6	0	GIO data direction of port n
GioDir5	0	GIO data direction of port n
GioDir4	0	GIO data direction of port n
GioDir3	0	GIO data direction of port n
GioDir2	0	GIO data direction of port n
GioDir1	0	GIO data direction of port n
GioDir0	0	GIO data direction of port n
DIn	0x00000000	Data Input Gio B [Memory Mapped]
DOut	0x00000000	Data Output Gio B [Memory Mapped]
DSet	0x00000000	Data Set Gio B [Memory Mapped]
DClr	0x00000000	Data Clear Gio B [Memory Mapped]
PDr	0x00000000	Open Drain Gio B [Memory Mapped]
PDis	0x00000000	Pull Disable Gio B [Memory Mapped]
PSel	0x00000000	Pull Select Gio B [Memory Mapped]